**ЛАБОРАТОРНАЯ РАБОТА №7**

**ШАБЛОНЫ ПРОЕКТИРОВАНИЯ**

Цель работы: ознакомиться с основными шаблонами проектирования, научиться

применять их при проектировании и разработке ПО.

Программное обеспечение: JDK 1.8+; IntelliJ IDEA; плагин PlantUML integration.

Вариант №4, 12, 20, 28

Шаблон “Фабричный метод”. Проект “Фабрика смартфонов”. В проекте должно быть реализовано создание смартфонов с различными характеристиками

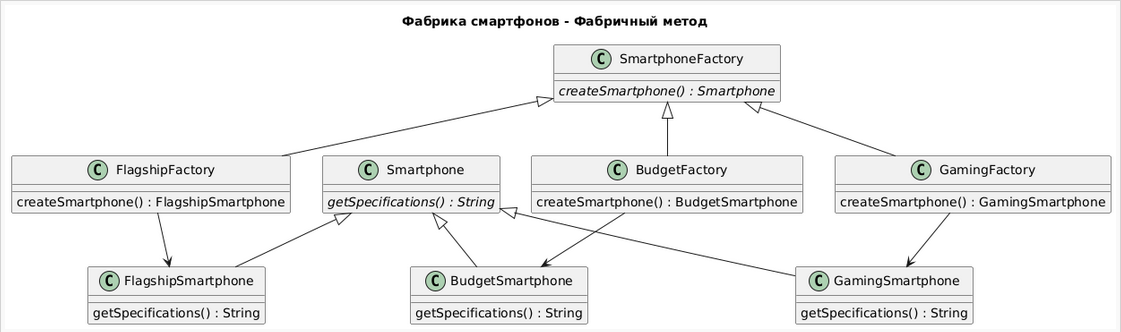


Рисунок 7.1 – **Диаграмма UML**

**Код Plantuml**

@startuml

title Фабрика смартфонов - Фабричный метод

!define AbstractFactory SmartphoneFactory

' Абстрактный класс смартфона (Product)

class Smartphone {

{abstract} getSpecifications() : String

}

' Конкретные продукты (ConcreteProduct)

class FlagshipSmartphone {

getSpecifications() : String

}

class BudgetSmartphone {

getSpecifications() : String

}

class GamingSmartphone {

getSpecifications() : String

}

Smartphone <|-- FlagshipSmartphone

Smartphone <|-- BudgetSmartphone

Smartphone <|-- GamingSmartphone

' Абстрактный класс фабрики (Creator)

class SmartphoneFactory {

{abstract} createSmartphone() : Smartphone

}

' Конкретные фабрики (ConcreteCreator)

class FlagshipFactory {

createSmartphone() : FlagshipSmartphone

}

class BudgetFactory {

createSmartphone() : BudgetSmartphone

}

class GamingFactory {

createSmartphone() : GamingSmartphone

}

SmartphoneFactory <|-- FlagshipFactory

SmartphoneFactory <|-- BudgetFactory

SmartphoneFactory <|-- GamingFactory

' Связь между фабриками и смартфонами

FlagshipFactory --> FlagshipSmartphone

BudgetFactory --> BudgetSmartphone

GamingFactory --> GamingSmartphone

@enduml

**Структура проекта**

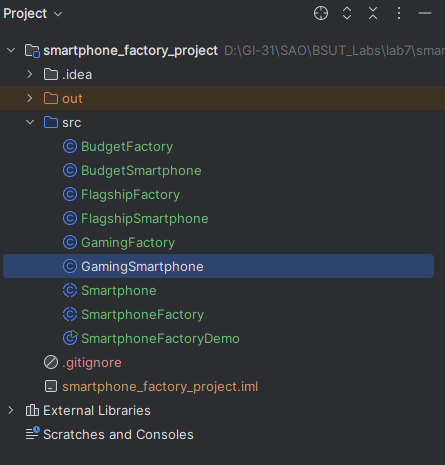
****

Рисунок 7.2 – **Структура проекта на Java**

**Код Java**

public class BudgetFactory extends SmartphoneFactory {  
 @Override  
 public Smartphone createSmartphone() {  
 return new BudgetSmartphone();  
 }  
}

public class BudgetSmartphone extends Smartphone {  
 @Override  
 public String getSpecifications() {  
 return "Model: Budget Y\nProcessor: Snapdragon 4 Gen 1\nRAM: 4GB\nStorage: 64GB\nCamera: 48MP Single Lens\nBattery: 4000mAh";  
 }  
}

public class FlagshipFactory extends SmartphoneFactory {  
 @Override  
 public Smartphone createSmartphone() {  
 return new FlagshipSmartphone();  
 }  
}

public class FlagshipSmartphone extends Smartphone {  
 @Override  
 public String getSpecifications() {  
 return "Model: Flagship X\nProcessor: Snapdragon 8 Gen 2\nRAM: 12GB\nStorage: 256GB\nCamera: 108MP Triple Lens\nBattery: 5000mAh";  
 }  
}

public class GamingFactory extends SmartphoneFactory {  
 @Override  
 public Smartphone createSmartphone() {  
 return new GamingSmartphone();  
 }  
}

public class GamingSmartphone extends Smartphone {  
 @Override  
 public String getSpecifications() {  
 return "Model: Gaming Z\nProcessor: MediaTek Dimensity 9200\nRAM: 16GB\nStorage: 512GB\nCamera: 64MP Dual Lens\nBattery: 6000mAh with fast charge";  
 }  
}

public abstract class SmartphoneFactory {  
 public abstract Smartphone createSmartphone();  
}

public abstract class Smartphone {  
 public abstract String getSpecifications();  
}

public class SmartphoneFactoryDemo {  
 public static void main(String[] args) {  
 // Создаем фабрики для разных типов смартфонов  
 SmartphoneFactory flagshipFactory = new FlagshipFactory();  
 SmartphoneFactory budgetFactory = new BudgetFactory();  
 SmartphoneFactory gamingFactory = new GamingFactory();  
  
 // Создаем смартфоны через фабрики  
 Smartphone flagship = flagshipFactory.createSmartphone();  
 Smartphone budget = budgetFactory.createSmartphone();  
 Smartphone gaming = gamingFactory.createSmartphone();  
  
 // Выводим характеристики смартфонов  
 System.*out*.println("Flagship Smartphone:\n" + flagship.getSpecifications());  
 System.*out*.println("\nBudget Smartphone:\n" + budget.getSpecifications());  
 System.*out*.println("\nGaming Smartphone:\n" + gaming.getSpecifications());  
 }  
}

**Результат работы**

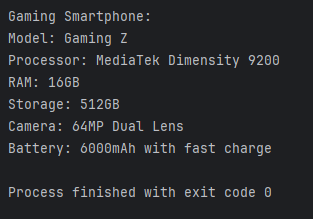


Рисунок 7.3 – **Результат работы проекта**